RISM-MOL-3D Documentation

Volodymyr P. Sergiievskyi

November 23, 2011

Contents

1 Theory

1.1 3D RISM

In the 3DRISM model which is used by the RISM-MOL-3D Solvent molecules are described in the 1DRISM approximation, while the solute molecule is a three-dimensional object. Structure of the

solvent is described by the total and direct correlation functions $h_\alpha(\mathbf{r})$, $c_\alpha(\mathbf{r})$ of the solute site α. The 3DRISM equations are written in the following way:

$$h_\alpha(\mathbf{r}) = \sum_{\xi=1}^{N_{solvent}} \int_{\mathbb{R}^3} c_\xi(\mathbf{r}) \chi_{\xi\alpha}(\mathbf{r}-\mathbf{r}) d\mathbf{r} \tag{1}$$

where N_{solvent} is the number of the solvent sites, $\chi_{\xi\alpha}(\mathbf{r})$ is the solvent susceptibility function for the sites ξ and α. Equation (1) is completed by the closure relations:

$$h_\alpha(\mathbf{r}) = e^{-\beta U_\alpha((r)) + h_\alpha(\mathbf{r}) - c_\alpha(\mathbf{r}) + B_\alpha(\mathbf{r})} - 1 \tag{2}$$

where $\beta = 1/k_B T$, k_B is the Boltzmann constant, T is the temperature, $U_\alpha(\mathbf{r})$ is the potential of the solute site α, $B_\alpha(\mathbf{r})$ is the bridge functional.

For the convenience of the iterative solution, the equations (1) are written in the following form [1]:

$$\gamma_\alpha(\mathbf{r}) = \sum_{\xi=1}^{N_{\text{solvent}}} \int_{\mathbb{R}^3} \mathcal{C}[\gamma_\xi(\mathbf{r}'-\mathbf{r})] \cdot \chi_{\xi\alpha}(\mathbf{r}') d\mathbf{r}' + \theta_\alpha(\mathbf{r}) - \mathcal{C}[\gamma_\alpha(\mathbf{r})] \tag{3}$$

where $\gamma_\alpha(\mathbf{r}) = h_\alpha(\mathbf{r}) - c_\alpha^S(\mathbf{r})$, $c_\alpha^S(\mathbf{r}) = c_\alpha(\mathbf{r}) + \beta U_\alpha^L(\mathbf{r})$, $U_\alpha(\mathbf{r}) = U_\alpha^S(\mathbf{r}) + U_\alpha^L(\mathbf{r})$, $U_\alpha^S(\mathbf{r})$ is the short range potential, $U_\alpha^L(\mathbf{r})$ is the long range potential, $\theta_\alpha(\mathbf{r}) = -\beta \sum_\xi \int_{\mathbb{R}^3} U_\xi^L(\mathbf{r}-\mathbf{r}') \chi_{\xi\alpha}(\mathbf{r}') d\mathbf{r}'$, $\mathcal{C}[\cdot]$ is the closure functional.

We use the interaction potentials which are superpositions of the site-site potentials:

$$U_\alpha^S(\mathbf{r}) = \sum_{s=1}^{N_{\text{solute}}} u_{s\alpha}^S(|\mathbf{r}-\mathbf{r_s}|) \tag{4}$$

$$U_\alpha^L(\mathbf{r}) = \sum_{s=1}^{N_{\text{solvent}}} u_{s\alpha}^L(|\mathbf{r}-\mathbf{r_s}|) \tag{5}$$

where $\mathbf{r}_s$ is the position of the solute site s with respect to the center of the molecule, N_{solute} is a number of the solute sites. The site-site potentials contain Lennard-Jones and Coulomb part. Pair Lennard-Jones parameters are obtained from the atom LJ parameters using the Lorentz-Berthelot mixing rules. To separate the short range and the long range potentials we use the Gauss Error Function [2].

Two different closures are implemented:

1) Hypper Netted Chain (HNC) closure:

$$\mathcal{C}[\gamma_\alpha(\mathbf{r})] = e^{-\beta U_\alpha^S(\mathbf{r}) + \gamma_\alpha(\mathbf{r})} - \gamma_\alpha(\mathbf{r}) - 1 \tag{6}$$

2) the Kovalenko-Hirata (KH) closure, which is defined as following [3]:

$$\mathcal{C}[\gamma_\alpha(\mathbf{r})] = \begin{cases} e^{-\beta U_\alpha^S(\mathbf{r}) + \gamma_\alpha(\mathbf{r})} - \gamma_\alpha(\mathbf{r}) - 1 & \text{when} \quad -\beta U_\alpha^S(\mathbf{r}) + \gamma_\alpha(\mathbf{r}) > 0 \\ -\beta U_\alpha^S(\mathbf{r}) & \text{otherwise} \end{cases} \tag{7}$$

In the numerical representation of equations (3) the functions $\gamma_\alpha(\mathbf{r})$, $\chi_{\xi\alpha}(\mathbf{r})$, $\theta_\alpha(\mathbf{r})$ are defined by their values in the grid points of an uniform Cartesian grid. The grid is defined by two parameters: *spacing* and *buffer*. Spacing is the distance between the grid points, and buffer is the minimal distance from the solute atoms to the boundaries of the grid (see Figure 1 for explanations).

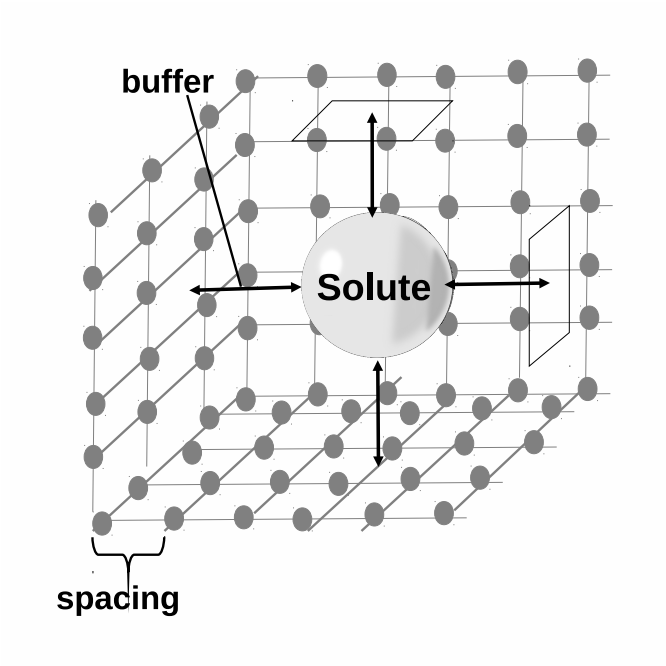

Figure 1: *Spacing* is the distance between the grid points, *buffer* is the minimal distance from the solute atoms to the boundaries of the grid

Using the same buffer parameter we can adjust the size and the shape of the grid preserving the constant cutoff of the solvent correlation functions for the different solutes. That provides us a straightforward way to control the accuracy of the calculations.

We denote the forward and inverse Fourier transforms on the grid $\mathcal{G}$ as $\mathcal{F}_{\mathcal{G}}[\cdot]$, $\mathcal{F}_{\mathcal{G}}^{-1}[\cdot]$ correspondingly. Then the discrete analogue of equations (3) reads as:

$$\mathbf{\Gamma}^{\mathcal{G}} = \mathcal{F}_{\mathcal{G}}^{-1}\left[\hat{\mathbf{X}} \cdot \mathcal{F}_{\mathcal{G}}\left[\mathcal{C}\left[\mathbf{\Gamma}^{\mathcal{G}}\right]\right]\right] + \mathbf{\Theta}^{\mathcal{G}} - \mathcal{C}\left[\mathbf{\Gamma}^{\mathcal{G}}\right] \tag{8}$$

where $\mathbf{\Gamma}^{\mathcal{G}} = \left(\boldsymbol{\gamma}_1^{\mathcal{G}}, \ldots, \boldsymbol{\gamma}_{N_{\text{solvent}}}^{\mathcal{G}}\right)^T$, $\mathbf{\Theta}^{\mathcal{G}} = \left(\boldsymbol{\theta}_1^{\mathcal{G}}, \ldots, \boldsymbol{\theta}_{N_{\text{solvent}}}^{\mathcal{G}}\right)^T$, $\hat{\mathbf{X}}^{\mathcal{G}} = [\boldsymbol{\chi}_{\boldsymbol{\xi}\boldsymbol{\alpha}}^{\mathcal{G}}]_{N_{\text{solvent}} \times N_{\text{solvent}}}$, upper index $\mathcal{G}$ means that functions are given by their values in the grid points $\mathcal{G}$.

Equation (8) can be written in a more compact way:

$$\mathbf{\Gamma}^{\mathcal{G}} = F[\mathbf{\Gamma}^{\mathcal{G}}] \tag{9}$$

where $F[\mathbf{\Gamma}^{\mathcal{G}}] = \mathcal{F}_{\mathcal{G}}^{-1}\left[\hat{\mathbf{X}} \cdot \mathcal{F}_{\mathcal{G}}\left[\mathcal{C}\left[\mathbf{\Gamma}^{\mathcal{G}}\right]\right]\right] + \mathbf{\Theta}^{\mathcal{G}} - \mathcal{C}\left[\mathbf{\Gamma}^{\mathcal{G}}\right]$.

Picard iteration method is defined by the recurrent formula:

$$\mathbf{\Gamma}_{n+1}^{\mathcal{G}} = (1-\lambda)\mathbf{\Gamma}_n^{\mathcal{G}} + \lambda F[\mathbf{\Gamma}_n^{\mathcal{G}}] \tag{10}$$

where $\mathbf{\Gamma}_n^{\mathcal{G}}$ is an n-th step approximation, λ is a coupling parameter.

1.2 DIIS and MDIIS iteration

Direct inverse in the iterative subspace (DIIS) method is an iteration method, initially introduced to improve convergence of the Schrödinger equation solvers [4]. Later, modified DIIS (MDIIS) method was applied to the solution of the 3DRISM equations [5]. In the DIIS method on the n-th

iteration step one finds a solution approximation $\mathbf{\Gamma}^{\mathcal{G}}_*$, which is a linear combination of approximations on the k previous iteration steps:

$$\mathbf{\Gamma}^{\mathcal{G}}_* = \sum_{i=1}^{k} C_i \mathbf{\Gamma}^{\mathcal{G}}_{n-k+i} \tag{11}$$

Below we describe the DIIS and MDIIS algorithms, which solve the generalized task:

$$\mathbf{\Gamma}^{\mathcal{G}} = F^{\mathcal{G}}[\mathbf{\Gamma}^{\mathcal{G}}] + \mathbf{D}^{\mathcal{G}} \tag{12}$$

where $\mathbf{D} = \left(\mathbf{d}^{\mathcal{G}}_1, \ldots, \mathbf{d}^{\mathcal{G}}_{N_{\text{solvent}}}\right)^T$ is a vector of corrections. [1]

Coefficients C_i in the formula (11) are chosen to minimize the norm of the residue $\Delta^{\mathcal{G}}_* = \mathbf{\Gamma}^{\mathcal{G}}_* - F[\mathbf{\Gamma}^{\mathcal{G}}_*] - \mathbf{D}^{\mathcal{G}}$. If one assumes linearity of the operator F, which for smooth operators is locally true, the task reduces to the system of linear equations[4]:

$$\begin{pmatrix} a_{11} & \ldots & a_{1k} & -1 \\ \vdots & \vdots & \vdots & -1 \\ a_{k1} & \ldots & a_{kk} & -1 \\ 1 & \ldots & 1 & 0 \end{pmatrix} \begin{pmatrix} C_1 \\ \vdots \\ C_k \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \tag{13}$$

where $a_{ij} = \int_{\mathbb{R}^3} \Delta_i(\mathbf{r})\Delta_j(\mathbf{r})d\mathbf{r}$, $\Delta_i(\mathbf{r}) = \mathbf{\Gamma}^{\mathcal{G}}_{n-k+i} - F[\mathbf{\Gamma}^{\mathcal{G}}_{n-k+i}] - \mathbf{D}^{\mathcal{G}}$. In the DIIS method $\mathbf{\Gamma}^{\mathcal{G}}_*$ is used as the solution approximation on the (n+1)-st iteration step. However, such procedure can lead to a linear dependent system of equations. MDIIS iteration method avoids this problem by adding some weighted residue to the (n+1)-st step of approximation [5]:

$$\mathbf{\Gamma'}^{\mathcal{G}} = \mathbf{\Gamma}^{\mathcal{G}}_* + \eta\left(F[\mathbf{\Gamma}^{\mathcal{G}}_*] + \mathbf{D}^{\mathcal{G}} - \mathbf{\Gamma}^{\mathcal{G}}_*\right) \tag{14}$$

where η is a weight for the residue. In the combination with the standard damping technique, a solution approximation on the (n+1)-st step $\mathbf{\Gamma}^{\mathcal{G}}_{n+1}$ in the MDIIS method can be written as following:

$$\mathbf{\Gamma}^{\mathcal{G}}_{n+1} = (1-\lambda)\Gamma^{\mathcal{G}}_n + \lambda\mathbf{\Gamma}^{\mathcal{G}}_* + \lambda\eta\left(F[\mathbf{\Gamma}^{\mathcal{G}}_*] + \mathbf{D}^{\mathcal{G}} - \mathbf{\Gamma}^{\mathcal{G}}_*\right) \tag{15}$$

In our work we use $\lambda = 0.5$, $\eta = 0.3$. To make notations shorter, we define MDIIS operator $\Xi[\cdot,\cdot]$ as following:

$$\Xi[\Gamma^{\mathcal{G}}_n, \mathbf{D}^{\mathcal{G}}] = (1-\lambda)\Gamma^{\mathcal{G}}_n + \lambda\mathbf{\Gamma}^{\mathcal{G}}_* + \lambda\eta\left(F[\mathbf{\Gamma}^{\mathcal{G}}_*] + \mathbf{D}^{\mathcal{G}} - \mathbf{\Gamma}^{\mathcal{G}}_*\right) \tag{16}$$

1.3 Multi-grid

We use the multi-grid technique in order to decrease the computation time spent on solving the 3DRISM equations. A description of the multi-grid theory can be found in the book [6]. Here we give only a short description of the multi-grid method applied to the 3DRISM equations. More information on the theoretical background of the method can be found in our recent paper, where similar computational framework for an efficient algorithm for solving the 1D-RISM equations is described [7].

In the multi-grid method the task is discretized on several grids with the same buffer but different spacing. Grids with the smaller number of the points and larger spacing are called *coarse*,

[1] It is useful to consider the generalized task (12) rather than the task (9) for description of the multi-grid algorithm.

grids with the larger number of the points and smaller spacing are called *fine*. In our work we consider the grids, whose number of points differ by the factor of 2^n, where $n = 0, 1, 2, ...$

We introduce operators $p[\cdot]$, $r[\cdot]$, which convert a coarse grid to a finer one and vice versa. We introduce the operator $R[\cdot]$, which move a fine-grid function to a coarse grid.

$$R[\mathbf{\Gamma}^{\mathcal{G}}] = \mathbf{\Gamma}^{r[\mathcal{G}]} \tag{17}$$

Also we introduce the operator $P[\cdot]$, which interpolates a coarse-grid function to a fine grid:

$$P[\mathbf{\Gamma}^{r[\mathcal{G}]}] = \mathbf{\Gamma}^{\mathcal{G}}_{\mathbf{1}} \tag{18}$$

In the paper we use the linear interpolation operator.

To make notations simpler, we introduce the operator $\Lambda_{\mathcal{G}}[\cdot;\cdot]$:

$$\Lambda_{\mathcal{G}}[\mathbf{\Gamma}_{\mathcal{G}}; \mathbf{D}^{\mathcal{G}}] = (1-\lambda)\mathbf{\Gamma}^{\mathcal{G}} + \lambda\left(F_{\mathcal{G}}[\mathbf{\Gamma}^{\mathcal{G}}] + \mathbf{D}^{\mathcal{G}}\right) \tag{19}$$

Multi-grid iterative algorithm, which solves the task (12), can be written in a following form:

$$\mathbf{\Gamma}^{\mathcal{G}}_{n+1} = \mathcal{M}^{l}_{\mathcal{G}}\left[\mathbf{\Gamma}^{\mathcal{G}}_{n}; \mathbf{D}^{\mathcal{G}}\right] \tag{20}$$

where $\mathbf{\Gamma}^{\mathcal{G}}_{n}$ is an n-th step approximation, $\mathcal{M}^{l}_{\mathcal{G}}[\cdot;\cdot]$ is the multi-grid operator, which performs one multi-grid iteration step of depth l on the grid $\mathcal{G}$. To calculate the multi-grid operator of depth $l = 0$ one performs m_0 one-grid iteration steps on the grid $\mathcal{G}$. Multi-grid technique can be applied to both: Picard and MDIIS iteration. We define the generalized operator $\Phi_{\mathcal{G}}[\cdot;\cdot]$ in a following way:

$$\Phi_{\mathcal{G}}[\mathbf{\Gamma}^{\mathcal{G}}_{n}; \mathbf{D}^{\mathcal{G}}] = \begin{cases} \Lambda_{\mathcal{G}}[\mathbf{\Gamma}^{\mathcal{G}}_{n}; \mathbf{D}^{\mathcal{G}}] & \text{for MG-Picard method} \\ \Xi_{\mathcal{G}}[\mathbf{\Gamma}^{\mathcal{G}}_{n}; \mathbf{D}^{\mathcal{G}}] & \text{for MG-MDIIS method} \end{cases} \tag{21}$$

Then the multi-grid operator of depth $l = 0$ is written as:

$$\mathcal{M}^{0}_{\mathcal{G}}\left[\mathbf{\Gamma}^{\mathcal{G}}; \mathbf{D}^{\mathcal{G}}\right] = \left(\Phi_{\mathcal{G}}\right)^{m_0}\left[\mathbf{\Gamma}^{\mathcal{G}}; \mathbf{D}^{\mathcal{G}}\right] \tag{22}$$

For $l > 0$, given n-th step approximation $\mathbf{\Gamma}^{\mathcal{G}}_{n}$ and correction vector $\mathbf{D}^{\mathcal{G}}$, the multi-grid operator $\mathcal{M}^{l}_{\mathcal{G}}[\cdot;\cdot]$ is calculated by the following algorithm:

Input parameters: $\mathbf{\Gamma}^{\mathcal{G}}_{n}$, $\mathbf{D}^{\mathcal{G}}$, l

Result: $\mathbf{\Gamma}^{\mathcal{G}}_{n+1} = \mathcal{M}^{l}_{\mathcal{G}}[\mathbf{\Gamma}^{\mathcal{G}}_{n}; \mathbf{D}^{\mathcal{G}}]$

1. Perform ν_1 Picard iteration steps on the fine grid (in our work $\nu_1 = 5$):

$$\mathbf{\Gamma'}^{\mathcal{G}} = \left(\Lambda_{\mathcal{G}}\right)^{\nu_1}\left[\mathbf{\Gamma}^{\mathcal{G}}_{n}; \mathbf{D}^{\mathcal{G}}\right]$$

2. Move to the coarse grid $r[\mathcal{G}]$:

$$\mathbf{\Gamma}^{r[\mathcal{G}]}_{(0)} = R[\mathbf{\Gamma'}^{\mathcal{G}}]$$

3. Calculate coarse-grid correction:

$$\mathbf{E}^{r[\mathcal{G}]} = R\left[F[\mathbf{\Gamma'}^{\mathcal{G}}]\right] - F[\mathbf{\Gamma}^{r[\mathcal{G}]}_{(0)}]$$

4. Perform recursively μ multi-grid iteration steps of depth $l-1$ on the coarse-grid (in our work μ=1):

$$\mathbf{\Gamma}^{r[\mathcal{G}]}_{(\mu)} = \left(\mathcal{M}^{l-1}_{r[\mathcal{G}]}\right)^{\mu}\left[\mathbf{\Gamma}^{r[\mathcal{G}]}_{(0)}; R[\mathbf{D}^{\mathcal{G}}] + \mathbf{E}^{r[\mathcal{G}]}\right]$$

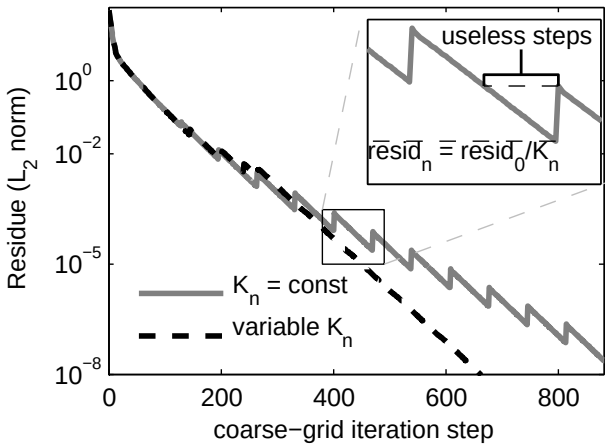

Figure 2: Coarse-grid residue decays with the number of the iteration steps in the multi-grid method. Two cases are shown: constant decay factor $K_n = 10$ (solid line), and variable decay factor K_n (dashed line). System: argon aqueous solution, spacing 0.1Å , buffer 6.4Å Peaks on the saw-shaped line (K_n=const) correspond to the boundaries of the multi-grid iteration steps. When iteration returns from the coarse grid to the fine grid, coarse-grid correction is re-calculated. Saw-shaped line means that iteration steps on the coarse grid are performed even after the accuracy of the coarse-grid correction calculation is reached. Thus, some number of coarse-grid iteration steps are idle, because they do not lead to the improving of the final result. Introducing a variable decay factor allows to adjust the accuracy of the coarse-grid calculations and avoid idle iteration steps.

5. Correct the fine-grid solution using the coarse-grid results:

$$\mathbf{\Gamma}''^{\mathcal{G}} = \mathbf{\Gamma}'^{\mathcal{G}} + P\left[\mathbf{\Gamma}^{r[\mathcal{G}]}_{(\mu)} - \mathbf{\Gamma}^{r[\mathcal{G}]}_{(0)}\right]$$

6. Perform ν_2 Picard iteration steps on the fine grid (in our work $\nu_2 = 0$):

$$\mathbf{\Gamma}^{\mathcal{G}}_{n+1} = (\Lambda_{\mathcal{G}})^{\nu_2}\left[\mathbf{\Gamma}''^{\mathcal{G}}; \mathbf{D}^{\mathcal{G}}\right]$$

The number of the iteration steps m_0 in the multi-grid operator of depth $l = 0$ depends on the number of the multi-grid iteration step n: $m_0 = m_0(n)$. We define $m_0(n)$ in such a way, that after the $m_0(n)$ iteration steps the residue decays by the factor K_n:

$$K_n||\left(\Phi_{\mathcal{G}}\right)^{m_0(n)}[\mathbf{\Gamma}^{\mathcal{G}}_n; \mathbf{D}^{\mathcal{G}}] - \left(\Phi_{\mathcal{G}}\right)^{m_0(n)+1}[\mathbf{\Gamma}^{\mathcal{G}}_n; \mathbf{D}^{\mathcal{G}}]|| < ||\mathbf{\Gamma}^{\mathcal{G}}_n - \Phi_{\mathcal{G}}[\mathbf{\Gamma}^{\mathcal{G}}_n; \mathbf{D}^{\mathcal{G}}]|| \tag{23}$$

We call the value K_n *the decay factor*.

Constant decay factor may lead to a non-smooth residue decay from a one multi-grid iteration step to another, which leads to the increasing of a number of the idle coarse-grid iteration steps (see Figure 2, solid line). To achieve a smoother error decay, in our paper we change the decay factor K_n by the following recursive formula:

$$K_{n+1} = \begin{cases} \max(\frac{1}{\alpha}K_n, K_{\min}) & \text{if} \quad ||\mathbf{\Gamma}^{\mathcal{G}}_{n,m_0} - \Phi_{\mathcal{G}}[\mathbf{\Gamma}^{\mathcal{G}}_{n,m_0}; \mathbf{D}^{\mathcal{G}}]|| < ||\mathbf{\Gamma}^{\mathcal{G}}_{n+1} - \Phi_{\mathcal{G}}[\mathbf{\Gamma}^{\mathcal{G}}_{n+1}; \mathbf{D}^{\mathcal{G}}]|| \\ \min(\beta K_n, K_{\max}) & \text{else} \end{cases} \tag{24}$$

where $\mathbf{\Gamma}^{\mathcal{G}}_{n,m_0} = (\Phi_{\mathcal{G}})^{m_0(n)}[\mathbf{\Gamma}^{\mathcal{G}}_n; \mathbf{D}^{\mathcal{G}}]$, $\alpha = 2$, $\beta = 1.2$. For the MG-Picard method we use $K_0 = 10$, $K_{\min} = 5$, $K_{\max} = 100$, for the MG-MDIIS method we use $K_0 = 100$, $K_{\min} = 10$, $K_{\max} = 100$.

This allows us to make the error decay smoother and to reduce a total number of iteration steps (see Figure 2, dashed line).

Usually, an iteration stops, when the norm of the residue is less than some threshold. However, this method has its own disadvantages. The first one is that the small residue does not immediately imply the small distance from the current approximation to the exact solution. The second one is that the threshold is usually given in dimensionless values, which are far from the physical meaning and thus one has no guidelines to chose an appropriate threshold. We use another criteria to stop iteration steps. Multi-grid iteration stops on the n-th iteration step, if the following condition is satisfied:

$$||\mathbf{\Gamma}_n - \mathbf{\Gamma}_{n+m}|| < \varepsilon_{\text{tres}} \tag{25}$$

where m is such, that

$$||\mathbf{\Gamma}^{\mathcal{G}}_{n+m} - \mathbf{\Gamma}^{\mathcal{G}}_{n+m+1}|| < 0.01||\mathbf{\Gamma}^{\mathcal{G}}_{n} - \mathbf{\Gamma}^{\mathcal{G}}_{n+1}|| \tag{26}$$

We use such condition, because usually $\mathbf{\Gamma}^{\mathcal{G}}_{n+m}$ is a good approximation of the exact solution. In the paper we use the norm, based on the Solvation Free Energy:

$$||\mathbf{\Gamma}^{\mathcal{G}}_{1} - \mathbf{\Gamma}^{\mathcal{G}}_{2}|| = |\Delta G_{KH}(\mathbf{\Gamma}_1) - \Delta G_{KH}(\mathbf{\Gamma}_2)| \tag{27}$$

where the solvation free energy is calculated in 3DRISM-KH approximation [8]:

$$\Delta G_{KH}(\mathbf{\Gamma}^{\mathcal{G}}) = \rho k_B T \sum_{\alpha}^{N_{\text{solvent}}} \int_{\mathbb{R}^3} \theta(-h_\alpha(\mathbf{r}))h_\alpha(\mathbf{r}) - \frac{1}{2}c_\alpha(\mathbf{r})h_\alpha(\mathbf{r}) - c_\alpha(\mathbf{r})d\mathbf{r} \tag{28}$$

Because of such definition a threshold has understandable physical meaning and is measured in energy units. In our work we use $\varepsilon_{\text{tres}}$=0.001 kcal/mol.

To make the calculations faster, in addition to the multi-grid technique we use several grids with the same spacing but different buffers. We introduce the grid-enlargement operator $e[\cdot]$, which enlarges the buffer of the grid. We introduce the operator $E[\cdot]$, which extrapolates a solution $\mathbf{\Gamma}^{\mathcal{G}}$ to the grid $e[\mathcal{G}]$.

$$E[\mathbf{\Gamma}^{\mathcal{G}}] = \mathbf{\Gamma}^{e[\mathcal{G}]} \tag{29}$$

Because the functions $\gamma_\alpha(\mathbf{r})$ tend to the zero when $|\mathbf{r}| \to \infty$, the operator $E[\cdot]$ simply extrapolates the functions by adding zeros at the parts of the grid $e[\mathcal{G}]$ which are not included into the grid $\mathcal{G}$. The scheme of the iteration can be written in a following way:

$$\mathbf{\Gamma}^{\mathcal{G}}_0 \xrightarrow{\text{solve 3DRISM eqs.}} \mathbf{\Gamma}^{\mathcal{G}}_* \xrightarrow{E[\cdot]} \mathbf{\Gamma}^{e[\mathcal{G}]}_0 \xrightarrow{\text{solve 3DRISM eqs.}} \mathbf{\Gamma}^{e[\mathcal{G}]}_* \xrightarrow{E[\cdot]} \dots \tag{30}$$

We start from the zero approximation $\mathbf{\Gamma}^{\mathcal{G}}_0$ on the grid $\mathcal{G}$ with the small buffer and using the scheme (30) after several steps obtain the solution on the grid with the large buffer.

1.4 Solvation Free Energy and Partial Molar Volume

There are several methods to calculate the solvation free energies. In the propam the following methods are implemented:

1. Hypper-Netted-Chain

$$\Delta G_{KH}(\mathbf{\Gamma}^{\mathcal{G}}) = \rho k_B T \sum_{\alpha}^{N_{\text{solvent}}} \int_{\mathbb{R}^3} -h^2_\alpha(\mathbf{r})/2 - \frac{1}{2}c_\alpha(\mathbf{r})h_\alpha(\mathbf{r}) - c_\alpha(\mathbf{r})d\mathbf{r} \tag{31}$$

2. Gaussian-Fluctuations [9]:

$$\Delta G_{GF} = \rho k_B T \sum_{\alpha=1}^{N_{\text{site}}} \int_{\mathbb{R}^3} (-\frac{1}{2} h_\alpha(\mathbf{r}) c_\alpha(\mathbf{r}) - c_\alpha(\mathbf{r})) d\mathbf{r} \tag{32}$$

3. Kovalenko-Hirata expression[8]:

$$\Delta G_{KH}(\mathbf{\Gamma}^{\mathcal{G}}) = \rho k_B T \sum_{\alpha}^{N_{\text{solvent}}} \int_{\mathbb{R}^3} \theta(-h_\alpha(\mathbf{r})/2) h_\alpha(\mathbf{r}) - \frac{1}{2} c_\alpha(\mathbf{r}) h_\alpha(\mathbf{r}) - c_\alpha(\mathbf{r}) d\mathbf{r} \tag{33}$$

To calculate the Partial Molar Volume the following formula is used[10, 11]:

$$V = \left(\frac{1}{\rho} + 4\pi \int_0^\infty (g_{oo}(r) - 1) r^2 dr \right) \left(1 - \rho \sum_{\alpha=1}^{N_{\text{site}}} \int_{\mathbb{R}^3} c_\alpha(\mathbf{r}) d\mathbf{r} \right) \tag{34}$$

where $g_{oo}(r)$ is the oxygen-oxygen RDF of the bulk water.

2 How to install

To install the program you will need to download and un-archive the files.

unzip RISM-MOL-3D.zip cd RISM-MOL-3D

If you have Intel-compatible 64 bit Linux system, than you are ready to use the program. Otherwise you will probably want to compile the program.

2.1 How to compile

To compile the program use the scripts ./makeMultiGrid and ./makeFreeEnergyLast:

./makeMultiGrid

./makeFreeEnergyLast

The source code depends on the LAPACK, BLAS, and FFTW3. The versions for 64bit linux are in the distributive. They are called: lapack_LINUX.a, blas_LINUX.a, libfftw3.a If you have another system, you will need to link appropriate libraries. Please, make necessary changes in the script file makeMultiGrid

Also you will need to compile the calculateFreeEnergyLast program To do this please use the script makeFreeEnergyLast. Make appropriate changes to the script, if necessary.

3 How to run the 3DRISM calculations

3.1 Files needed to run

The following files are needed to run the calculations:

- *Solute Structure File* - solute molecule description
- *Solvent File* - solvent atoms (sites) description

- *Solvent RDFs file* - RDFs functions of the bulk solvent
- *Solvent Omega File* - solvent omega functions $\omega_{\alpha\xi}(r)$ description
- *Parameters File* - file where all necessary parameters are given

See details of the file formats in the sections below

3.2 The command to run the 3DRISM calculations

To run the 3DRISM calculations use the command
./multiGridMain parameters.txt
Optiannaly you can use the command
./multiGridMain parameters.txt 'SomeExtaParameters'
Here parameters.txt is the *Parameters File*, *SomeExtraParameters* are the additional parameters for the calculation, separated by the ';'
Example:
./multiGridMain parameters.txt 'SolventStructureFile="Ar.rism"; NDIIS=6;buffer=8[Angstr];
'

3.3 The output files

The following files are produced:

- *g functions* - they are called NameOfSolute_in_NameOfSolvent_gNumber.3d where NameOfSolute is the name of the solute file, NameOf Solvent is the name of the solvent file, Number is the number of the solvent site
- *c functions* - they are called NameOfSolute_in_NameOfSolvent_cNumber.3d
- *gamma functions* - they are called NameOfSolute_in_NameOfSolvent_gammaNumber.3d
- *grid points in X,Y,Z directions* - NameOfSolute_in_NameOfSolvent_X.grd, NameOfSolute_in_NameOfSolv NameOfSolute_in_NameOfSolvent_Z.grd
- *grid points in the Fourier Space in X,Y,Z directions* - NameOfSolute_in_NameOfSolvent_Kx.grd, NameOfSolute_in_NameOfSolvent_Ky.grd, NameOfSolute_in_NameOfSolvent_Kz.grd

See details of the file formats in the sections below

3.4 How to visualize the results?

To visualize the results you may use the matlab. The following link can be useful for you:
http://compchemmpi.wikispaces.com/Visualisation#Sergiievskyi3D_1

4 How to calculate the Partial Molar Volume and the Solvation Free Energy

1. if not compiled - compile the file calculateFreeEnergyLast

   ./makeFreeEnergyLast

2. Run the command ./calculateFreeEnergyLast

   ./calculateFreeEnergyLast Solute_in_Solvent_ parameters.txt 'SomeExtraParameters'

   Here Solute is the name of solute file (without extention) Solvent is the name of the solvent file (without extention)

   parameters.txt is the file with parameters used in RISM-MOL-3D

   SomeExtraParameters are the extra parameters, obtained from the simulations. Typically SomeExtraParameters are: 'SoluteStructureFile="Solute.rism";'

See also examples of calculations in the sections below

5 Example: DO ALL

5.1 Do the 3DRISM calculations

To run the 3DRISM calculations for the argone in water do the following:
Go to the 3DRISM installation folder. Run the commands:
./multiGridMain parameters.txt 'SoluteStructureFile="Ar.rism";'
The following files will be produced:
Ar_in_water_g0.3d - g function of oxygen
Ar_in_water_g1.3d - g function of hydrogen
Ar_in_water_c0.3d - c function of oxygen
Ar_in_water_c1.3d - c function of hydrogen
Ar_in_water_gamma0.3d - gamma function of oxygen
Ar_in_water_gamma1.3d - gamma function of hydrogen
Ar_in_water_X.grd - grid in the X direction
Ar_in_water_Y.grd - grid in the Y direction
Ar_in_water_Z.grd - grid in the Z direction
Ar_in_water_Kx.grd - grid in the Fourier space in X direction
Ar_in_water_Ky.grd - grid in the Fourier space in Y direction
Ar_in_water_Kz.grd - grid in the Fourier space in Z direction

5.2 Visualize results

To visualize results run the following commands in matlab (you should be in the directory where the RISM-MOL-3D is installed) The scripts also work in GNU-octave, however, due to partial incompatibility of plots in octave and matlab the results are much worse.
gridX=load('Ar_in_water_X.grd'); gridY=load('Ar_in_water_Y.grd'); gridZ=load('Ar_in_water_Z.grd');
g0=load('Ar_in_water_g0.3d');
[g3D,GX,GY,GZ,X,Y,Z]=prepareToSlice(g0,gridX,gridY,gridZ,{'z','y','x'});

```
h=slice(GX,GY,GZ,g3D,[],[],[0]);
set(h,'LineStyle','none')
colorbar
hold on
System=load('Ar.rism');
XYZ=System(:,1:3)/0.52;
Sigma=System(:,4)/0.52;
Color=ones(length(Sigma),3)*0.5;
shl=plot_mol2(XYZ,Sigma,Color,10);
xlabel('X [Bohr]'); ylabel('Y [Bohr]'); set(h,'FaceLighting','none');
```

5.3 Run the Solvation Free Energy and Partial Molar Volume calculations

./calculateFreeEnergyLast Ar_in_water_ parameters.txt 'SoluteStructureFile="Ar.rism";'

The 4 last line in the output are: - the SFE calculated using the HNC expression - the SFE calculated using the KH expression - the SFE calculated using the GF expression - the Partial Molar Volume

6 Format of Files

6.1 Solute Structure File

SoluteStructureFile describes the structure of the solute. it contain 6 columns (each line corresponds to one atom):

X,Y,Z,sigma,epsilon,charge
X,Y,Z are coordinates of atoms
sigma and epsilon are lennard-jones parameters
charge is a partial charge of atom

6.2 Solvent File

SolventDataFile describes the solvent atoms it contains 5 columns (each line corresponds to one atom): sigma, epsilon, charge, multiplicity, density sigma and epsilon are Lennad-Jones parameters charge is a partial charge of atom multiplicity defines, how many identical sites are there in the solvent molecule density defines the density of each of sites

6.3 Solvent RDFs file

The file contains the RDFs of the solvent. Default units: Bohrs. The file consists of the columns of numbers

1st column - grid points 2nd column - g11,
3rd column - g12,
...
(N-1 + 2) column - g1N

N +2 column - g22
N+1 +2 column - g23
...
(2N-2 + 2) column - g2N
(2N-1 +2) column - g33
...
N(N-1)/2 +2 column- gNN

where N is number of unique solvent sites, gIJ means RDF between sites I and J

6.4 Solvent Omega File

File, which describes the omega functions of different sites each line corresponds to one omega function The file contain 3 columns: RDFnumber, omega_koefficient, omega_shift RDFnumber describes the number of RDF to which the omega function corresponds where RDFs are numbered in the following order:

0 - g11,
1 - g12,
...
N-1 - g1N
N - g22
N+1 - g23
...
2N-2 - g2N
2N-1 - g33
...
N(N-1)/2 - gNN

where N is number of unique solvent sites, gIJ means RDF between sites I and J
the resulting omega function will be pre-multiplied by the omega_koefficient
omega_shift defines the position of the delta sphere
Thus, result is omega_koefficient*delta(—r-omega_shift—)

6.5 Parameters File

Parameters file consists of the records.

Each record looks like this: Parameter = Value [Units]; (the symbol ';' is absolutely necessary!!!!)

The file may also have the comments, which begin from the '#' sign. The comment is the whole-line

The typical content of the parameters.txt file is presented below:

GRID

Buffer = 15 [Angstr];
spacing = 0.2 [Angstr];

ITERATION

```
#iteration. valid values: 'Picard' or 'DIIS' (case sencentive)
iteration = 'MDIIS';

#Number of DIIS vectors (only usedm then iteration='DIIS')
NDIIS = 5;

#damping of MDIIS iterations
# x' = (1 - preDamping) * x[n] + preDamping * sum(alpha[i]*x[i])
# x[n+1] = (1 - postDamping) * x' + postDamping * F[x']
diisPreDamping = 1;
diisPostDamping = 0.3;

# Should we use multi grid? Valid values: 'yes' or 'no'
MultiGrid = 'yes';

#if yes, what is the multi grid depth?
# (depth means, who much times the number of grid points will be fivided by two.
# e.g. for the initial grid 128x128x128 with dx=dy=dz=0.1 [Angstr] and depth=2
# the coarse grid is 64x64x64 with dx=dy=dz=0.4 [Angstr]
depth = 3;

# Number of extentions.
# After the main cycle of iteration converge, the solutions will be prolongated
# to the larger grid, and iterations repeat.
# this parameters regulates the number of extentions.
# e.g. for initial grid 128x128x128 with dx=dy=dz=0.1[Angstr] and number of extentions =2
# the resultin grid is 512x512x512 with dx=dy=dz=0
NumberOfExtentions = 2;

# Number Of pre smooting steps in multi-grid iteration
NumberOfPreSmoothingSteps = 5; # DO NOT CHANGE

#Number of post smoothing
NumberOfPostSmoothingSteps =0; # DO NOT CHANGE

#NumberOfMultiGridSubSteps. 1 - V iteration 2- W iteration
NumberOfMultiGridSubSteps = 1; # DO NOT CHANGE

# initial lambda coupling parameter
lambda = 0.5;

# norm which is used to control convergence of the iterations
# Valid values are: 'L2' and 'HNC'
norm = 'HNC';

# decayFactor regulates, who often we calculate the norm.
```

# let x[i] are the solutions, and n is the smallest natural, such, that
# —x[n]-x[n+1]— ¡ —x[0]-x[1]—/ DecayFactor
# then the norm is calculated as —x[n]-x[0]—
# (this allows to calculate norm more precicely, because —x[n]-x[0]— is a better
# approximation to —x*-x[0]—, than —x[n]-x[n+1]—
DecayFactor=100;
#decayFactor = 10;

# if after fine-grid correction next coarse error is less than before correction, that means that not enough
# steps was done (error of grid correction is ¡ than coarse error).
# The decay factor will be multiplied by DecayFactorMultiplier, and thus number of coarse iteration steps increase
DecayFactorMultiplier = 1.2;

# if after fine-grid correction next coarse error is more than before correction, that means that too much
# steps was done (error of grid correction is ¿ than coarse error).
# The decay factor will be divided by DecayFactorMultiplier, and thus number of coarse iteration steps decrease
DecayFactorDivider = 2;

#Minimal and maximal limits for decayFactor
#MinDecayFactor = 1.1;
MinDecayFactor = 10;
MaxDecayFactor = 100;

# tolerance which is used to stop the iterations.
# iteratins stop, then norm (see decayFcator) is less than tolerance
# value of tolrance depeds on the norm used and on the decay factor
tolerance = 0.001 [kcal/mol];

DIVERGENCE

#growFactor controls the condition to detect the divergence of iterations
# when the norm of difference between sequential solutions grows more than
# by growFactor times, the iteration is supposed to diverge.
growFactor = 100;

first grow factor shows, how much times should the error grow to start divergence check (RealNormCondition::stateDiverge)
#firstGrowFactor = 1;
FirstGrowFactor = 1e9;

divergenceLambdaDecayFactor.
when the iteration diverge, the value of lambda is decreasing by divergenceLambdaDecayFactor,

```
and iteration steps are continued with the new lambda
divergenceLambdaDecayFactor = 10;

# Not more than MaxCoarseStepCount coarse grid steps is allowed on the coarse grid
MaxCoarseStepCount = 1000;

# If coarse grid norm reaches MinCoarseError, iteration steps stop
MinCoarseError = 1e-13;


######### SOLUTE

# SoluteStructureFile describes the structure of the solute.
# it contain 6 columns (each line corresponds to one atom):
# X,Y,Z,sigma,epsilon,charge
# X,Y,Z are coordinates of atoms
# sigma and epsilon are lennard-jones parameters
# charge is a partial charge of atom
SoluteStructureFile = "Ar.rism";

#Distance units, which are used in SoluteStructureFile for X,Y,Z and sigma
SoluteDistanceUnits = "Angstr";

#Energy units, which are used in SoluteStructureFile for epsilon
SoluteEnergyUnits = "kcal/mol";

######## SOLVENT

# SolventDataFile describes the solvent atoms
# it contains 5 columns (each line corresponds to one atom):
# sigma, epsilon, charge, multiplicity, density
# sigma and epsilon are Lennad-Jones parameters
# charge is a partial charge of atom
# multiplicity defines, how many identical sites are there in the solvent molecule
# density defines the density of each of sites
SolventDataFile = "water.slv";

#Distance units, which are used in SolventDataFile for sigma
SolventDistanceUnits = "Angstr";

#Energy units, which are used in SolventDataFile for epsilon
SoventEnergyUnits = "kcal/mol";

#File, which contains bulk-solvent radial distribution functions
#Structure of file:
# 1st column - regular grid in R direction
# next N * (N+1)/2 columns - samples of RDF functions
```

```
# where N is number of unique solvent sites
# order is g11 g12 ... g1N g22... g2N g33 ... gNN
# where gIJ means RDF between I-th and J-th solvent sites
SolventRDFsFile = "waterRDFs.txt";

#Distance units, which are used for the first column in SolventRDFsFile
SolventRDFsUnits = "Bohr";

#File, which describes the omega functions of different sites
# each line corresponds to one omega function
# The file contain 3 columns:
# RDFnumber, omega_koefficient, omega_shift
# RDFnumber describes the number of RDF to which the omega function corresponds
# where RDFs are numbered in the following order:
# 0 - g11,
# 1 - g12,
# ...
# N-1 - g1N
# N - g22
# N+1 - g23
# ...
# 2N-2 - g2N
# 2N-1 - g33
# ...
# N(N-1)/2 - gNN
#
# where N is number of unique solvent sites,
# gIJ means RDF between sites I and J
#
# the resulting omega function will be pre-multiplied by the omega_koefficient
#
# omega_shift defines the position of the delta sphere
#
# Thus, result is omega_koefficient*delta(—r-omega_shift—)
SolventOmegaFile = "waterOmega.txt";

# Distance units used in the SolventOmegaFile
OmegaDistanceUnits = "Angstr";

#Units which are used for the density (the last column) in the solventDayaFile
SolventDensityDistanceUnits = "nm";


#Temperature of the solvent
T = 300 [ K ]; # do not forget to put [K] after the value!!!

####### POTENTIALS AND CLOSURE
```

```
# closure. Valid values are 'HNC' and 'KH'
closure = 'KH';

# mixing rules. Valid Values are 'LornentzBerthelot' and 'OPLSAA'
# LorentzBerthelot means sigma12 = (sigma1+sigma2)/2, epsilon12=sqrt(epsilon1*epsilon2)
#OPLSAA means sigma12=sqrt(sigma1*sigma2); epsilon12=sqrt(epsilon1*epsilon2);
MixingRules = 'LorentzBerthelot';

# Mixing coefficient for ng-procedure (multiplier of the argument of erf function ).
# Determines, how smooth is the transition from the short-range to thelong-range function (the smaller - the smoother).
ngCoeff = 0.5;


# Output energy units (for Free Energy file)
OutputEnergyUnits = 'kcal/mol';
```

References

[1] J. S. Perkyns, G. C. Lynch, J. J. Howard, and B. M. Pettitt. Protein solvation from theory and simulation: Exact treatment of coulomb interactions in three-dimensional theories. *Journal of Chemical Physics*, 132(6):064106, February 2010.

[2] K. C. Ng. Hypernetted chain solutions for the classical one-component plasma up to /*gamma* =7000. *Journal of Chemical Physics*, 61(7):2680–2689, 1974.

[3] A. Kovalenko and F. Hirata. Potentials of mean force of simple ions in ambient aqueous solution. II. Solvation structure from the three-dimensional reference interaction site model approach, and comparison with simulations. *Journal of Chemical Physics*, 112(23):10403–10417, June 2000.

[4] P. Pulay. Convergence acceleration of iterative sequences - the case of scf iteration. *Chemical Physics Letters*, 73(2):393–398, 1980.

[5] A. Kovalenko, S. Ten-No, and F. Hirata. Solution of three-dimensional reference interaction site model and hypernetted chain equations for simple point charge water by modified method of direct inversion in iterative subspace. *Journal of Computational Chemistry*, 20(9):928–936, July 1999.

[6] W. Hackbusch. *Multi-grid methods and Applications*. Springer-Verlag, Berlin, 1985.

[7] V. P. Sergiievskyi, W. Hackbusch, and M. V. Fedorov. Multigrid solver for the reference interaction site model of molecular liquids theory. *Journal of Computational Chemistry*, 32(9):1982–1992, 2011.

[8] F. Hirata, editor. *Molecular theory of solvation*. Kluwer Academic Publishers, Dordrecht, Netherlands, 2003.

[9] D. Chandler, Y. Singh, and D. M. Richardson. Excess electrons in simple fluids .1. general equilibrium-theory for classical hard-sphere solvents. *Journal of Chemical Physics*, 81(4):1975–1982, 1984.

[10] T. Imai, M. Kinoshita, and F. Hirata. Salt effect on stability and solvation structure of peptide: An integral equation study. *Bulletin of the Chemical Society of Japan*, 73(5):1113–1122, May 2000.

[11] T. Imai, Y. Harano, A. Kovalenko, and F. Hirata. Theoretical study for volume changes associated with the helix-coil transition of peptides. *Biopolymers*, 59(7):512–519, December 2001.